Bulletin of Geophysics and Oceanography Vol. XX, n. X, pp. XXX-XXX; Xxxxx 20xx
DOI 10.4430/bgo00510

YOLOv8 based real-time underwater fish monitoring framework: detection, tracking, and counting

R. Garg and A.C. Phadke

Department of Electrical and Electronics Engineering, Dr. Vishwanath Karad MIT World Peace University, Pune, India

(Received: 8 October 2024; accepted: 13 October 2025; published online: 20 January 2026)

ABSTRACT YOLO (You Only Look Once) is one of the most popular computer vision algorithms. Computer vision has revolutionised the field of moving object detection in real time with its ability to analyse and understand visual content much like a human. This paper presents a comprehensive framework for fish detection, bounding box-based tracking, and counting in underwater environments using the YOLOv8 deep learning architecture. Accurately and efficiently identifying, tracking, and counting fish plays an important role in aquatic research, conservation efforts, and fishery management. The proposed system uses a pre-trained YOLOv8 model which is fine tuned using a large annotated dataset of underwater fish images. The model is improved using transfer learning to learn features specific to fish detection in water. Real-time underwater fish detection is performed on underwater video streams using a fine-tuned YOLOv8 model. The high speed and accuracy of YOLOv8 enables efficient localisation of fish instances at every frame. The analysis of such data enables accurate fish counts and facilitates effective monitoring and assessment of fish populations in water bodies. The true positive rate of 0.91 and accuracy of 92% indicated that the system successfully identified a significant proportion of fish instances present in the images.

Key words: computer vision, deep learning, object detection, real time object level tracking.

1. Introduction

Exploration and comprehension of marine ecosystems are now more crucial than ever for resource management and environmental preservation. Important indicators of the health of an ecosystem are fish population dynamics and behaviour. Manual surveys and acoustics are two common traditional approaches for fish detection and tracking in underwater habitats (Sprague *et al.*, 2014). However, when employing camera-based systems for fish monitoring, a crucial design consideration is the long-standing dichotomy between baited and unbaited systems. Baited remote underwater video stations can enhance detection rates by attracting fish, but they may introduce behavioural biases due to artificial attraction (Burgi, 2021; Knausgård *et al.*, 2022). In contrast, unbaited systems provide a more accurate representation of natural fish behaviour, though often at the cost of reduced detection rates (Coro and Walsh, 2021). The choice between these setups significantly impacts both the system design and the generalisability of AI-based solutions, as discussed in prior reviews (Barbedo, 2022). Moreover, traditional approaches are labour-intensive, time-consuming, and inaccuracy-prone. This underscores the growing need for automated, scalable, and accurate systems to facilitate effective fish population monitoring (Gruszczyński *et al.*, 2020).

He *et al.* (2016) highlighted the benefits of using deep residual networks for addressing the vanishing gradient problem and improving the accuracy of computer vision tasks. However, a number of difficulties arise when directly implementing these methods in aquatic settings. Water turbidity, uneven lighting, and colour distortion cause underwater photos to have poor visibility, which can have a big impact on how well traditional object detection algorithms work (Buono *et al.*, 2021). Additionally, the complexity and variety of underwater scenes, including the presence of corals, rocks, and other marine organisms, increases the possibility of occlusions and confusion in fish detection.

Redmon *et al.* (2016) introduced the original YOLO (You Only Look Once) algorithm, provided an overview of its architecture, and demonstrated its real-time object detection capabilities for computer vision applications. The study by Cimino *et al.* (2018) emphasises the importance of real-time detection for various applications in marine exploration, including species identification, habitat mapping, and environmental monitoring. There are many well-known deep learning approaches for object detection, including YOLO, Single Shot MultiBox Detector, Faster Region-based Convolutional Neural Network (R-CNN), Region-based Fully Convolutional Networks, RetinaNet, Mask R-CNN, etc. While each algorithm has its merits, YOLO stands out as a powerful solution for various reasons.

One key advantage of YOLO is its real-time processing capabilities. By adopting single-pass architecture, YOLO converts the input image into a grid form and simultaneously predicts and draws bounding boxes, thus mentioning probabilities for each class. This parallel processing enables YOLO to achieve faster inference speeds compared to other algorithms, making it well suited for real-time applications (Redmon *et al.*, 2017). Its ability to strike a balance between accuracy, speed, and robustness to occlusions makes it a compelling choice for practical applications. YOLO exhibits strong generalisation capabilities. Its effectiveness in underwater environments makes it particularly valuable for tasks such as fish detection in real time. The study by Fang *et al.* (2020) presented a comparative analysis of object detection algorithms for underwater robotic applications. The researchers evaluated the performance of different algorithms, including YOLO, in terms of real-time processing, robustness to occlusions, and accuracy in complex underwater scenes. The results show that YOLO achieves a real-time processing speed of 30 frames per second (FPS) with an accuracy of 92.5% which is better than other algorithms. In an article by Rath and Gupta (2023), a comprehensive analysis of various YOLO object localisation and detection models, which focused on their performance, in terms of FPS and Mean Average Precision (mAP) on different hardware setups, was presented. The article focused on the performances of the YOLOv5, YOLOv6, and YOLOv7 models. In terms of Graphics Processing Unit (GPU) performance, the study reveals that the YOLOv5 nano model stands out by achieving an impressive 230 FPS on an NVIDIA RTX 4090 GPU. All YOLO models in the study can run in real time on GPUs, with none dropping below 30 FPS. This demonstrates the effectiveness of YOLO models in leveraging GPU capabilities for fast and efficient object detection.

YOLOv8 (Jindal, 2023) introduces Darknet-53, a new backbone network that is considerably quicker and more precise than the one utilised in YOLOv7. A convolutional neural network (CNN) called DarkNet-53, which has 53 layers, is capable of classifying photos into thousands of different categories, including pencil, keyboard, mouse, and numerous animals. YOLOv8 generates detection box predictions in a manner akin to pixel-wise picture segmentation. Additionally, YOLOv8 uses a much bigger feature map and also a more effective CNN than earlier versions, making it more effective than earlier versions. As a result, the model can detect things more quickly and accurately, it can identify patterns and moving objects in the data in a much better way, and can capture more intricate interactions between various features with a larger

feature map. A larger feature map can also aid in accelerating the model's training and learning process and help to prevent overfitting. In addition to object detection, object tracking and counting play a vital role in computer vision applications, offering significant advantages. Object counting provides valuable quantitative information about the number of objects present in a scene (Khan et al., 2018).

2. Literature survey

The study by Sah *et al.* (2017) illustrates the distinction between object detection and object tracking within a video stream. It highlights that, in scenarios with occlusions, tracking techniques utilise temporal information to predict the position of an object. Amraee *et al.* (2022) conducted a study focusing on the efficiency of feature extraction methods for classifying small metal objects. In their research, they compared the Histogram of Oriented Gradients (HOG) and Local Binary Pattern (LBP) approaches with deep learning methods, including the popular YOLO algorithm. The authors examined the effectiveness of these algorithms in classifying screws, nuts, keys, and coins. While traditional feature extraction methods like HOG and LBP demonstrated efficiency in generating accurate feature vectors for classification, YOLO stood out as a deep learning-based approach capable of both object detection and classification.

Zhang *et al.* (2016) addressed the challenge of fish detection and tracking in underwater visual sensor networks using traditional computer vision techniques. They proposed a method that combines background subtraction and motion analysis for fish detection. However, these traditional approaches have limitations in handling complex underwater conditions, such as low visibility and colour distortion. Yousif and Ghareeb (2019) proposed a real-time fish detection and tracking method. They utilised background subtraction and contour-based feature extraction to detect fish in video frames. While their approach showed promising results in controlled underwater environments, it could face challenges in high dynamic backgrounds where the fish and background exhibit similar motion patterns. Ren *et al.* (2016) introduced Faster R-CNN, a state-of-the-art architecture for real-time object detection. This approach utilises region proposal networks to generate object proposals and, then, classifies and refines them. This method may require adaptations to handle the unique challenges posed by underwater environments.

Redmon and Farhadi (2018) proposed YOLOv3, an incremental improvement of previous versions of the YOLO architecture. YOLOv3 divides the input image into a grid and directly predicts bounding boxes and class probabilities. Its performance in underwater fish detection can be further improved by incorporating adaptations to handle low visibility, colour distortion, and changing light conditions. Li *et al.* (2017) explored fish counting and tracking in underwater video sequences using deep learning techniques. They proposed a combination of CNNs and recurrent neural networks to detect and track fish. Qin *et al.* (2020) proposed a robust visual tracking method for underwater fish using deep learning. They employed a Siamese network architecture to track fish across consecutive frames. Their method demonstrated robustness to changes in fish appearance and background clutter.

Pagire and Phadke (2020) achieved an accuracy of 99.74% using the mobile net model of a deep neural network to detect and recognise fish species. Though the trained model detected nine different types of fish species with high accuracy, the dataset used in the work is comparatively less complex with only one or two fishes in each image. In the work by Pagire and Phadke (2020), a Gaussian mixture model-based Godbehere-Matsukawa-Goldberg algorithm is used to detect moving objects in three types of backgrounds. The authors considered three

classes of backgrounds: static, moderate dynamic, and high dynamic backgrounds. The algorithm performance is comparatively poor while detecting objects in high dynamic backgrounds. The study by Lekunberri *et al.* (2022) focuses on the application of computer vision and deep learning techniques to identify and measure tropical tuna species in purse seiner catches and also presents a methodology to automatically classify and quantify different tuna species from captured images. In their work, Garg and Phadke (2024) address the pressing need for effective underwater fauna monitoring by conducting a comprehensive comparative study of two state-of-the-art object detection models: YOLOv4 and YOLOv8. Their research focuses on real-time fish detection and tracking within dynamic underwater environments, emphasising the importance of accuracy, processing speed, and adaptability to challenging conditions. Their findings demonstrate the superiority of the YOLOv8 model over the YOLOv4. In response to the challenges of underwater surveillance, Pagire and Phadke (2024) present a novel hybrid model for fish detection in aquatic environments. Central to this approach is the integration of the local structure binary pattern method with a unique attribute extractor called Multi Frame Triplet Pattern (MFTP). The MFTP enhances object detection by encoding background layout information across three successive frames, while considering local intensity variances.

This paper suggests reliable fish detection, bounding box-based tracking, and counting system based on YOLOv8. The real-time performance and high detection accuracy of YOLOv8 make it ideal for underwater monitoring applications. By implementing a number of significant improvements and adaptations, the performance of YOLOv8, in the underwater realm, is improved.

3. Methodology

Fig. 1 depicts the step-by-step process followed to accomplish underwater fish detection, bounding box-based tracking, and counting using the YOLOv8 algorithm. The proposed flow for the system involves annotating and pre-processing data, fine-tuning the YOLOv8 model with configured hyper parameters and augmented data and model evaluation using a separate test set to assess performance.

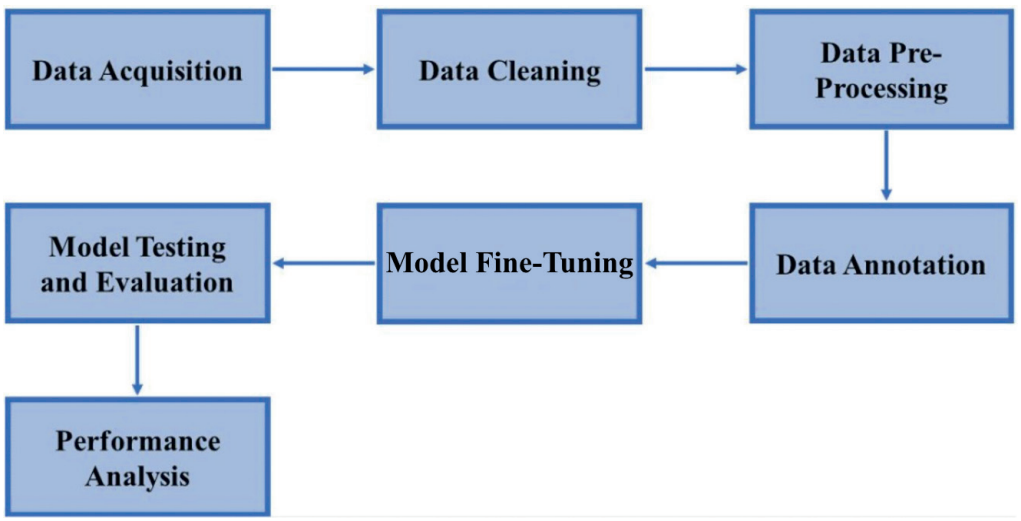

Fig. 1 - System block diagram.

3.1. Data acquisition

For this model a custom dataset, specifically curated for underwater fish monitoring, was created. The dataset comprises a collection of underwater images captured from various aquatic environments, including rivers, lakes, and oceans. The dataset consists of 1,000 images, each with an average size of 1024×768 pixels. For the construction of this dataset, multiple sources were utilised, the primary source being the Google Open Images dataset V7 (Kuznetsova *et al.*, 2020), which provided a diverse collection of underwater images. In addition, manual capture of aquarium images and videos incorporated controlled underwater environments and specific fish species.

3.2. Data cleaning

To ensure the suitability of the dataset for this research, relevance and quality, a rigorous selection process was implemented. Various unwanted images that did not align with the research purpose, such as those with low visibility, images outside water, animated fish images or those with insufficient fish presence, were removed from the dataset.

3.3. Data pre-processing and data augmentation

The pre-processing steps, performed on the acquired dataset for the proposed model, included contrast enhancement, noise reduction, and resizing of images to a standardised resolution. Different data augmentation techniques were used to enhance the diversity of the dataset, improve generalisability of the model, and avoid overfitting. Techniques such as random scaling, channel shifting, rotation, image distortion, and image saturation were employed to augment the dataset.

3.4. Data annotation

Accurate and reliable annotations serve as the foundation for training machine learning models, enabling them to learn and generalise from labelled data. Good annotations provide the necessary ground truth information, enabling the models to recognise and differentiate fish instances from the background (Joshi and Shivalker, 2023). The overall performance, robustness, and accuracy of the trained models are usually influenced by the quality of the annotations. Precise bounding box annotations help the models accurately localise fish objects within the images or videos. For the proposed work, the process of data annotation and labelling was conducted manually using the Computer Vision Annotation Tool (CVAT) (CVAT Documentation, 2023). CVAT is a versatile and user-friendly annotation platform that facilitates the efficient and precise labelling of objects in images and videos. The annotation results are provided in the ‘0 x_min y_min width height’ format, where 0 indicates the fish class label and, as this is a single class classification, all labels are labelled as 0. The coordinates (x_min, y_min) represent the normalised top-left corner of the bounding box, while the ‘width’ and ‘height’ represent its size. These annotations demonstrate the precise localisation and size estimation of fish objects within the images, aiding in object detection tasks and further analysis.

While horizontal bounding boxes provide a practical method for annotating fish locations, they do not capture the true orientation or shape of the fishes. This can introduce biases, especially in tracking scenarios where fish overlap or appear at oblique angles. In such cases, bounding boxes may contain significant background or intersect with other instances, leading to identity switches

or false detections. Moreover, this approach may limit biological insights related to behaviour, directionality, or body posture. Future work could explore oriented bounding boxes or instance segmentation to more accurately capture fish geometry and improve tracking fidelity.

3.5. Model fine-tuning

In the proposed work, the YOLOv8n (nano) model is selected as it offers its lightweight architecture, fast inference speed, and compatibility with constrained hardware. Ultralytics (Rath, 2023), a leading technology company in the field of computer vision and deep learning, is revolutionising the way we interact with visual data. The Ultralytics team has created an open-source software library called YOLOv8 that builds upon the success of the original YOLO algorithm. YOLOv8 offers different model variants with varying sizes, such as YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. These variants have different model complexities, enabling users to choose an appropriate balance between the computational efficiency and accuracy of a model, based on their specific requirements. Table 1 compares the different variants of the YOLOv8 model based on their specifications. The models have a fixed input size of 640 pixels and performance ranging from 37.3 to 53.9 mAPval, indicating their object detection accuracy. These specifications provide insights into the trade-offs between accuracy, speed, model size, and computational requirements for the YOLOv8 variants. The model underwent a comprehensive fine-tuning procedure, spanning 150 epochs, utilising a batch size of 16. The process lasted approximately three hours, leveraging the computational resources of the Google Colab platform. Throughout the process, input images of 640 pixels were utilised, ensuring an optimal balance between computational efficiency and detection accuracy. To evaluate model performance and generalisation, the dataset was divided into three distinct sets: training, validation, and testing, split in a ratio of 700:150:150, respectively, to ensure a sufficient number of samples for training, model selection, and final evaluation.

Table 1 - Different variants of the YOLOv8 model.

Model	Size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Parameters (M)	FLOPS (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Among the five YOLOv8 variants (n, s, m, l, and x), the YOLOv8n model was selected due to its lightweight architecture and compatibility with real-time processing on limited hardware. This model was fine-tuned on the curated underwater fish dataset.

3.5.1. Model selection

For the proposed work, the YOLOv8n model is preferred as it offers several advantages that align well with the requirements of the research. First and foremost, one of the primary reasons

for selecting YOLOv8n is its real-time inference capabilities and faster training capabilities. Additionally, YOLOv8n exhibits remarkable accuracy, even while operating at high speed. The model leverages a deep CNN architecture combined with advanced techniques like anchor free detections, multi-scale predictions, and feature pyramid networks. These elements enable it to efficiently locate and detect objects across different scales and aspect ratios, resulting in more precise bounding box predictions and improved overall performance. Considering the real-time inference capabilities, high accuracy, deployment flexibility, and strong community support, the YOLOv8n model emerged as the ideal choice for the proposed work.

3.5.2. Model architecture and specifications

The YOLOv8n model employed in this research work boasted an impressive architecture with 225 layers, comprising a total of 3,011,043 parameters. The deep structure of the model enabled it to capture intricate features and learn complex representations. With 3,011,027 gradients, the model had a considerable capacity for adjusting its parameters during the fine-tuning process. The YOLOv8n model demonstrated a computational efficiency of 8.2 gigaFLOPS, enabling rapid inference. Fig. 2 showcases the sequential arrangement of various layers involved in the process. Starting with convolutional (Conv) layers for feature extraction, the model incorporates C2f blocks—the lightweight residual CSP-style feature-extraction modules used throughout the backbone/neck—to improve gradient flow and representation capacity. Spatial Pyramid Pooling–Fast (SPPF) layers capture multi-scale information, while Upsample layers increase the spatial resolution for refined object localisation. Concatenation (Concat) layers combine features from different scales, and the final Detect layer generates bounding box predictions and class probabilities.

The backbone is responsible for the initial feature extraction from the input data. It typically consists of a series of Conv and pooling layers, and other operations. The primary objective of the backbone network is to capture and encode low-level and mid-level features present in the input data, while the head is responsible for taking the extracted features from the backbone and performing specific tasks, such as classification, regression, or object detection. It typically consists of additional Conv layers, and specialised modules tailored for the specific task at hand. The head network inputs the high-level features captured by the backbone and transforms these features into the desired output format. This comprehensive arrangement of Conv, C2f, SPPF, Upsample, Concat, and Detect layers enables YOLOv8 to effectively extract features, handle objects of different sizes, and produce accurate object detections. The YOLOv8 architecture utilises the Leaky Rectified Linear Unit (ReLU) activation function. This function is a modified version of the traditional ReLU function that also introduces a small slope/value for any negative input values, preventing the issue of "dying" neurons. In YOLOv8, the Leaky ReLU activation function is applied after certain layers to introduce nonlinearity and enable the network to learn complex representations. The Leaky ReLU function is defined as:

$$f(x) = (\alpha x, x)\,. \tag{1}$$

The graph of the Leaky ReLU function in Fig. 3 depicts that the function returns x if it receives any positive input, but for any negative value of x, it returns a very small value which is α times x. α is a small constant that determines the slope of the function for negative inputs. Typically, the value of α is set between 0.1 or 0.01 to ensure that there is a small non-zero gradient for negative inputs. The use of the Leaky ReLU activation function in YOLOv8 aids in capturing and

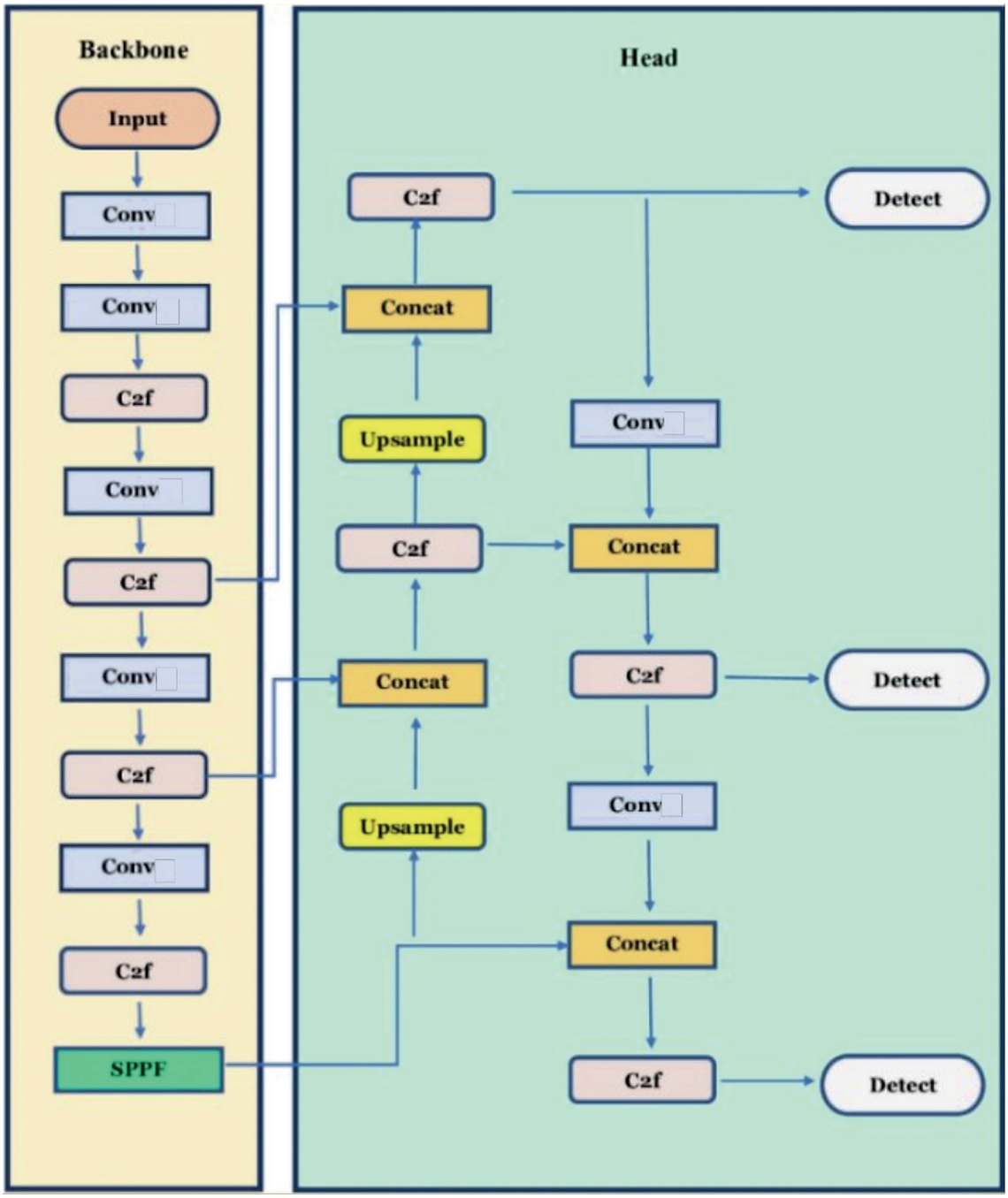

Fig. 2 - The YOLOv8 model architecture.

representing complex features and patterns in the input data.

The YOLOv8 architecture employs a combination of different loss functions to train the model and optimise its performance. The primary loss function used in YOLOv8 consists of three components:

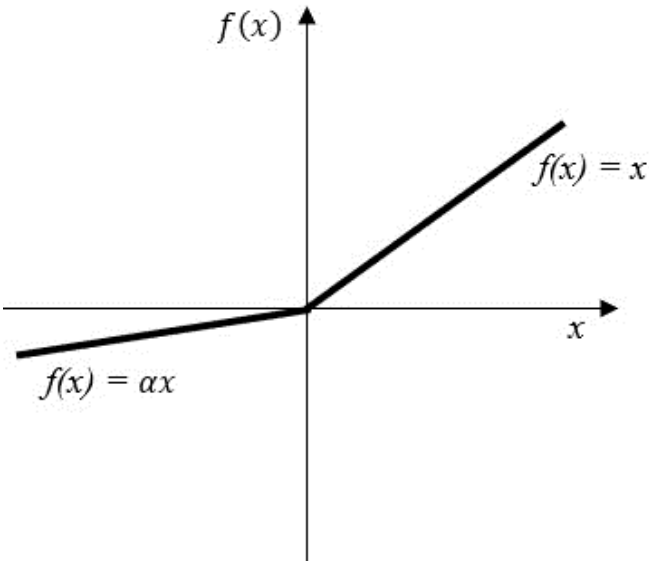

Fig. 3 - The Leaky ReLu activation function.

the box loss, the Distribution Focal Loss (DFL), and the class loss. The box loss is responsible for measuring the accuracy of predicted bounding box coordinates. It ensures that the predicted boxes closely align with the ground truth boxes. The box loss is measured using the Mean-Squared Error (MSE) loss. It penalises the discrepancies between the predicted and true box coordinates, encouraging precise localisation of objects. The MSE is used to calculate the average squared difference between the predicted and true values. In the case of box loss, it quantifies the error in the predicted box coordinates, allowing the model to learn to accurately localise objects. The formula for the MSE is:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2}$$

where N is the total number of bounding boxes, y_i represents the predicted box coordinates, and $\hat{y}_i$ represents the ground truth box coordinates.

The class loss evaluates the accuracy of predicted class probabilities. It is computed based on the binary cross-entropy loss for the confidence scores of each and every predicted bounding box, which encourages the model to assign high probabilities to the correct object classes and low probabilities to incorrect object classes. The binary cross-entropy loss is calculated as:

$$L_{cls} = -\frac{1}{n}\sum_{i=1}^{n} y_i log\left(\hat{y}_i\right) + \left(1 - y_i\right) log\left(1 - \hat{y}_i\right) \tag{3}$$

where L_{cls} is the binary cross-entropy loss, n represents the total number of bounding boxes, y_i represents the true label, and $\hat{y}_i$ represents the predicted probability/confidence score.

The DFL is a variant of the focal loss that helps improve model performance when training data is imbalanced. Specifically, the DFL is used to deal with class imbalance that arises when

training on datasets with very rare objects. The DFL aims at addressing this problem, thus making sure that the model correctly detects these rare objects.

3.5.3. Detection, bounding box-based tracking, and counting

For object detection, the YOLOv8n model was used. This model demonstrated high accuracy and efficiency in detecting objects of interest within images. The YOLOv8n architecture, with its 225 layers and 3,011,043 parameters, proved instrumental in capturing intricate features and learning complex representations.

To enable the real-time bounding box-based fish tracking system, the bounding box predictions from YOLOv8n were processed using the ByteTrack (Skalski, 2023) tracking algorithm. ByteTrack performs tracking-by-detection by associating detection boxes with tracklets, i.e. sequences of frames where an object has been continuously observed. Its core strength lies in a two-stage data association process. In the first stage, high-confidence detections are matched to existing tracklets to ensure reliable identity assignment and in the second stage, low-confidence detections, which are typically discarded by traditional trackers, are re-evaluated and matched using Intersection over Union (IoU) and appearance similarity metrics. The detailed working of the ByteTrack tracking-by-detection model is described here following.

D_t represents detections at frame *t* and T_{t-1} represents active tracks from the previous frame.

Step 1. Detections are split into high-confidence (D_t^H) and low-confidence (D_t^L) sets based on confidence thresholds τ_h and τ_L, using:

$$D_t^H = \{d \in D_t \mid s(d) > \tau_h\}, \quad D_t^L = \{d \in D_t \mid \tau_L \leq s(d) < \tau_h\}. \tag{4}$$

Step 2. D_t^H is matched to existing tracks using IoU:

$$IoU(bi, bj) = \quad | bi \cup bj | \; / \; | bi \cap bj |. \tag{5}$$

Step 3. For unmatched tracks, matching is performed with D_t^L using a combination of IoU and appearance similarity S_{app}:

$$\text{Match score} = \alpha \cdot (\text{IoU}) + (1 - \alpha) \cdot S_{app}. \tag{6}$$

This two-stage comprehensive matching strategy allows ByteTrack to recover missed or briefly occluded objects and maintain identity consistency across frames. A gating mechanism further filters redundant matches, enhancing robustness. However, ByteTrack assumes reasonably consistent motion and visual appearance and may experience identity switches in crowded scenes with overlapping fish.

For object counting, the Supervised learning Matlab library was used to implement a line-crossing strategy. A virtual counting line was drawn across the video frame, and each tracked fish was counted once upon crossing it, based on its unique tracker ID [refer to Eq. (7)]. This minimised false positives and avoided duplicate counts. For the line-crossing counting model, L is the virtual counting line with normal vector n, p_{t-1} and p_t represents the consecutive positions of a tracked fish, and $p_{L:}$ represents any point on the line.

A fish is counted when:

$$sign\ ((p_{t-1} - p_L) \bullet n) \neq sign\ ((p_t - p_L) \bullet n). \quad (7)$$

In simple terms: if a fish's path crosses from one side of the line to the other, the count is incremented once per unique ID.

By combining the detection capabilities of YOLOv8n, the tracking system built upon the YOLOv8 architecture using the Bytetrack package, and counting functionality implemented through the supervision package based on YOLOv8, the research achieved comprehensive detection, tracking, and counting capabilities.

3.5.4. Adaptations and modifications

To optimise YOLOv8 for underwater fish detection, several modifications were introduced to the model architecture. Firstly, the backbone network was fine tuned to handle underwater-specific image characteristics, such as low visibility and colour distortion. The anchor box design was modified to account for the size and shape variations of underwater fish species. Anchor boxes are predefined bounding boxes with a specific shape and size. In object detection algorithms like YOLO, anchor boxes are used to predict and localise objects in an image. During training, the model learns to predict the bounding box coordinates relative to the anchor box for each object present in the grid cell. This is done by regressing the offset values for the box coordinates. The predicted box coordinates, combined with the anchor box parameters, determine the final bounding box for the detected object. Fig. 4 illustrates the concept of anchor boxes in object detection models like YOLOv8. It depicts a grid overlaying the image, dividing it into multiple cells.

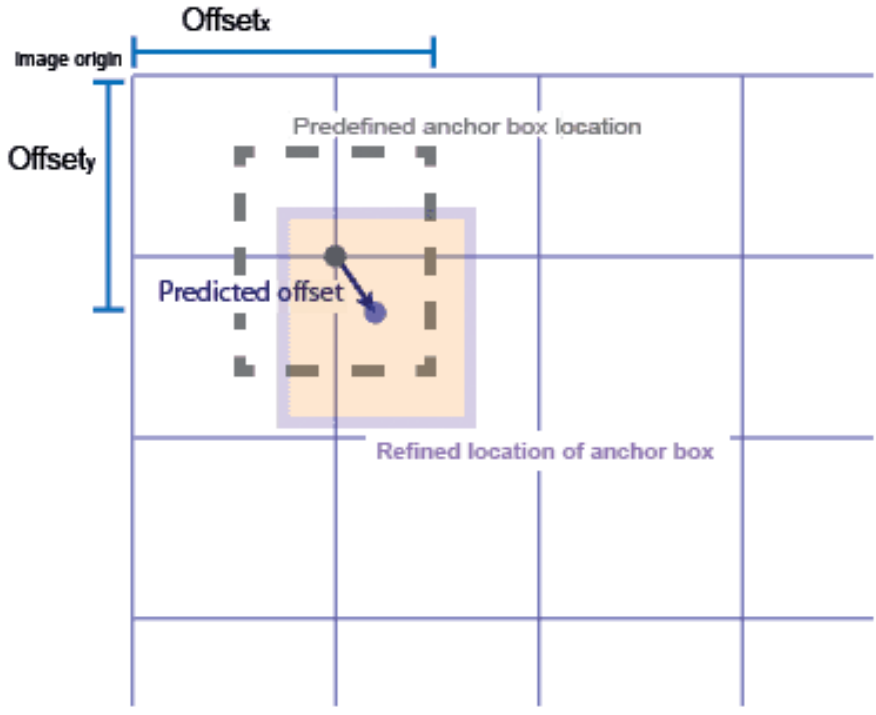

Fig. 4 - Anchor boxes for object detection (Mathworks, 2023).

Each cell is associated with anchor boxes, which are predefined bounding boxes of different sizes and aspect ratios. Furthermore, the YOLOv8 detection head was adjusted to handle the detection of fish in different orientations and postures. The adapted YOLOv8 model was evaluated using appropriate performance metrics, specifically tailored for underwater fish monitoring. These performance measures included F1 score, recall, mAP, and overall detection accuracy.

4. Results and discussion

The implementation of the system has yielded impressive outcomes. While multiple optimisations and iterations were performed during the system's development to enhance its performance, a total of 150 epochs were used to achieve optimal results and address specific challenges encountered in fish detection.

4.1. Graphical analysis of model performance

The resulting confusion matrix provides an overview of true positives, false positives, true negatives, and false negatives. In this case, 91% of fish instances were correctly detected during fine-tuning. The high true positive rate of 0.91 indicates that the system is effective in detecting fish instances. This suggests that the system has a good capability to identify and locate fish objects within underwater imagery. The low false negative rate of 0.09 implies that the system has a relatively low tendency to miss actual fish instances. Fig. 5 provides a label correlogram, which is a group of two-dimensional (2D) histograms showing each axis of data against the other. The labels in the images are provided in the x, y, width, height (xywh) space. The xywh representation is a common format used in object detection tasks to describe the bounding box coordinates of objects within an image. Here, x and y denote the coordinates of the top-left corner of the bounding box, while width and height represent the dimensions of the box.

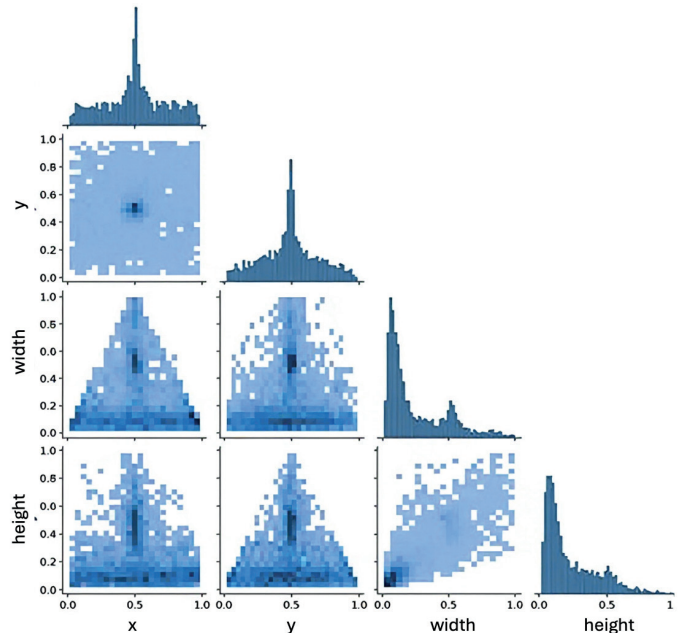

Fig. 5 - Label correlogram.

This format allows for precise localisation and description of objects; it also allows for 2D localisation and statistical characterisation of detected objects within the image. The histograms represent the distribution of fish bounding box sizes (width and height) and their relative positions (x and y coordinates) in the frame. These insights help in understanding where fishes are typically located and what their relative size is in terms of number of pixels by rectangular approximation. We are approximating fish shape by a rectangular bounding box rather than determining the precise outline of a fish through delineation of a smooth contour. Since the objective is to detect, track, and count fish, the bounding box approach provides a reliable estimate of fish numbers while incurring significantly lower computational cost. In contrast, active contours, snake algorithms, and other fine-grained segmentation or boundary-detection methods require iterative optimisation and high-resolution per-pixel processing, which leads to considerably greater computational demands (Meimetis *et al.*, 2025).

For the proposed application, such segmentation-based methods pose several practical drawbacks. First, they are inherently slower, often unsuitable for real-time or high-throughput scenarios, especially when dealing with video streams or large volumes of underwater imagery. Second, their performance can degrade in challenging underwater conditions (such as low contrast, motion blur, occlusion, or suspended particles) where precise contour delineation becomes difficult and may not yield a substantial accuracy gain over simpler methods. Third, segmentation outputs are often more sensitive to noise and require post-processing (mask refinement, morphological operations) to be usable for tracking and this further increases latency.

On the other hand, bounding box-based detectors can achieve comparable detection and counting accuracy for well-separated objects, as the objective is not to extract exact morphology but to identify and enumerate targets. Modern object detectors such as YOLO or Faster R-CNN generate bounding boxes in a single forward pass, avoiding the iterative convergence steps needed in classical segmentation. This results in a much lower computational overhead and faster inference speeds, enabling near real-time processing even on resource-constrained hardware. In practice, this trade-off, which consists in slightly reduced shape fidelity for a large gain in processing speed, makes bounding boxes an efficient and robust choice for fish detection, tracking, and counting in operational settings.

To evaluate the system's performance at different confidence levels, metrics such as precision, recall, and F1 score were computed across a range of confidence thresholds. Unlike standard classification tasks, where recall is typically reported at a single threshold, object detection systems produce confidence scores for each predicted bounding box. Varying this confidence threshold directly affects which detections are accepted or rejected. Fig. 6 presents the recall-confidence curve, which shows how recall changes as the confidence threshold increases. This curve illustrates the trade-off between recall and model confidence. Lower thresholds may yield higher recall but include more false positives, while higher thresholds improve precision at the cost of recall. Eq. (8) provides the recall formula used in this analysis:

$$Recall = \frac{TP}{(TP + FN)}. \tag{8}$$

Fig. 7 illustrates the precision confidence curve, which showcases the system's precision values for different confidence intervals. This provides an indication of the system's accuracy in correctly identifying fish instances. It is calculated by:

$$Precision = \frac{TP}{(TP + FP)}. \tag{9}$$

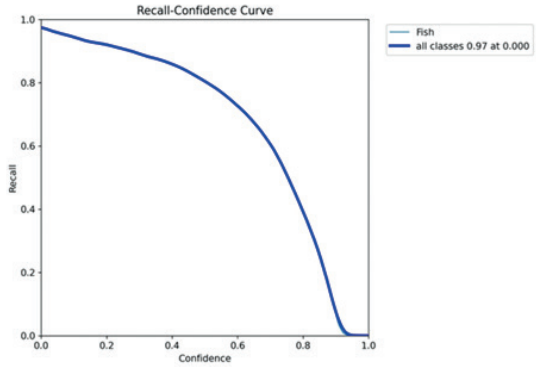

Fig. 6 - Recall confidence curve.

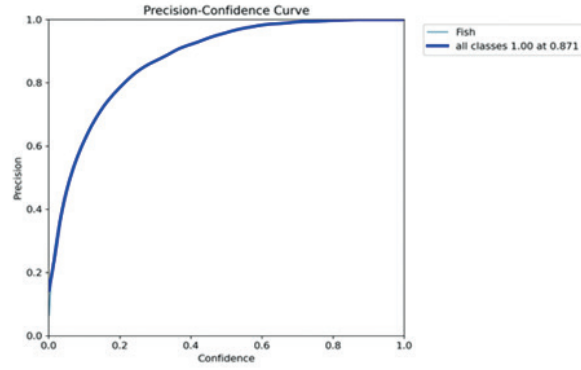

Fig. 7 - Precision confidence curve.

The F1-confidence curve, shown in Fig. 8, highlights the F1 scores achieved by the system at different confidence intervals. This metric combines precision and recall, providing an overall assessment of the system's performance. The best F1 score of 0.89 was observed at the 0.382 confidence interval. The F1 score is calculated by:

$$F1\ Score = \frac{2PR}{(P + R)}. \tag{10}$$

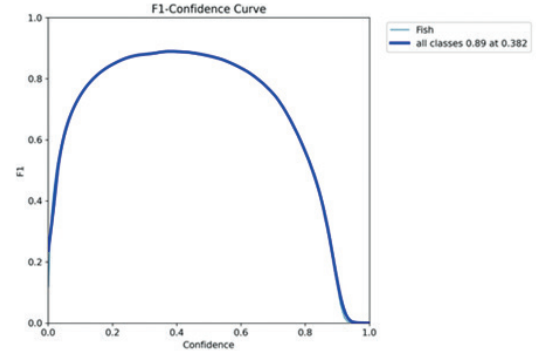

Fig. 8 - F1-confidence curve.

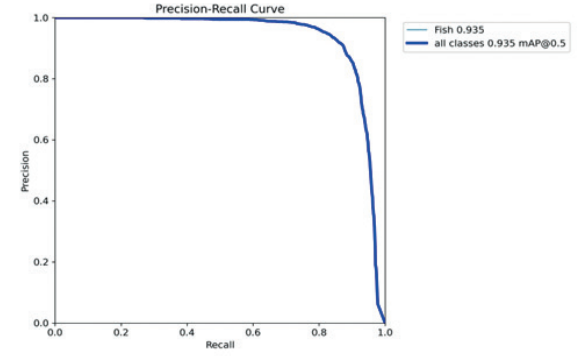

Fig. 9 - Precision recall trade-off curve.

This result indicates that the system attains a satisfactory blend of detection accuracy and completeness in fish detection. Additionally, the precision-recall trade-off curve is depicted in Fig. 9, providing insights into the trade-off between precision and recall for different confidence thresholds. By analysing the precision-recall trade-off curve, a best decision threshold, balancing precision and recall, was selected. Fig. 10 illustrates the graphical outputs and results obtained during the fine-tuning and evaluation of the YOLOv8 model. The figure includes 10 distinct

graphs providing a comprehensive overview of the convergence, loss functions, precision, recall, and mAP. The training and validation box losses show a convergence trend below 1.5 after 150 epochs.

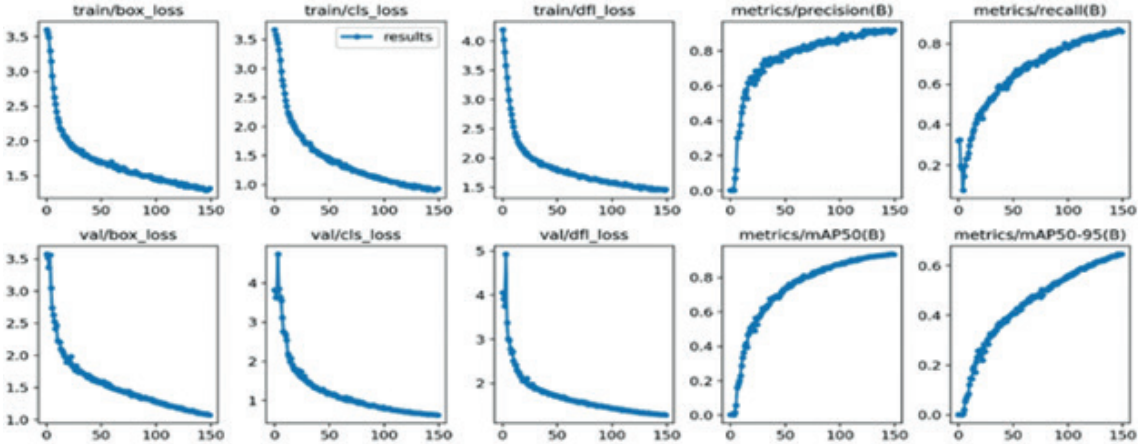

Fig. 10 - Performance analysis of YOLOv8 for underwater fish detection.

This indicates that the model has effectively minimised the localisation error and improved the accuracy of predicted bounding boxes. The training and validation class losses converged at around 1 after 150 epochs. This suggests that the model has successfully learnt to allocate higher probabilities to correct object classes and lower probabilities to incorrect classes, resulting in more accurate class predictions. The training and validation DFLs converge below 1.5, proving that the model is effectively handling the class imbalance and improving the detection performance.

The precision graph demonstrates the precision of the model's detections for the fish class. A high precision value, above 0.8, indicates that the model is making accurate fish detections most of the time. The recall graph shows the recall of the model's predictions for the fish class. A high recall value, above 0.8, indicates that the model is effectively capturing most of the fish instances in the dataset, thus minimising false negatives. The mAP50 graph represents the mAP at an IOU threshold of 0.5. It measures the overall detection performance of the model by considering both precision and recall across different IOU thresholds. IOU is calculated as the ratio of the intersection area to the union area of the two boxes. A mAP50 value between 0.8 and 1 indicates that the model is achieving high accuracy in detecting fish instances. The mAP50 value is calculated by:

$$mAP@0.5 = \frac{\left(precision@0.5 + precision@0.75 \; + \; precision@0.95\right)}{3}. \tag{11}$$

The mAP for IOU thresholds ranging from 0.5 to 0.95 is depicted by the mAP50-95 graph. It offers a more thorough assessment of the model's detection abilities over a wider variety of IOU thresholds. A mAP50-95 value between 0.6 and 0.8 suggests that the model maintains good detection performance across different levels of bounding box overlap. The mAP50-95 value is calculated by:

$$mAP@0.5-0.95=\frac{\left(precision@0.5+precision@0.55\ +\ldots+precision@95\ \right)}{10}. \quad (12)$$

Table 2 includes the model performance metrics such as recall, F1 score, the best confidence interval, and the overall average accuracy.

Table 2 - Summary of model results: performance metrics.

Performance measure	Value
F1 score	0.89
Recall	0.91
Best confidence interval	0.382
Overall average accuracy of the model	92%

4.2. Fish detection, bounding box tracking, and counting results

Figs. 11 and 12 present the detection outputs obtained from two variations of the model: one using only detection capabilities and the other incorporating detection, bounding box tracking, and counting functionalities. These figures clearly illustrate the improved performance of the latter configuration. The results also serve as practical demonstrations of the efficient operation of the ByteTrack tracking algorithm and the supervision-based line-crossing counting strategy discussed in section 3.5.3. The accurate maintenance of fish identities across frames and accurate counts exemplify the robustness of this integrated pipeline. Furthermore, the detection outputs were categorised into five types, each showcasing the model's performance under different conditions: number of fishes in a frame, visibility, similarity with the background, fish size, and fish position. These categories provide a comprehensive evaluation of the system's capabilities and highlight its potential to address diverse challenges in underwater fish monitoring.

Fig. 11 - Results of the detection model.

Fig. 12 - Results of the enhanced model.

Accuracies from these different scenarios served as a quantitative measure to assess the model's effectiveness under different conditions:

1. number of fishes in a frame. As per the number of fishes in the frame, the dataset is divided into three categories: fish count less than 3, fish count between 3 and 10, and fish count greater than 10, as shown in Fig. 13a, 13b, and 13c, respectively. Accuracy is

determined separately for these three categories and it is depicted in the first row of Table 3. From the table it is clear that the highest accuracy of 98% is obtained when the fish count is less than 3. As the number of fishes in a frame increases, accuracy decreases;

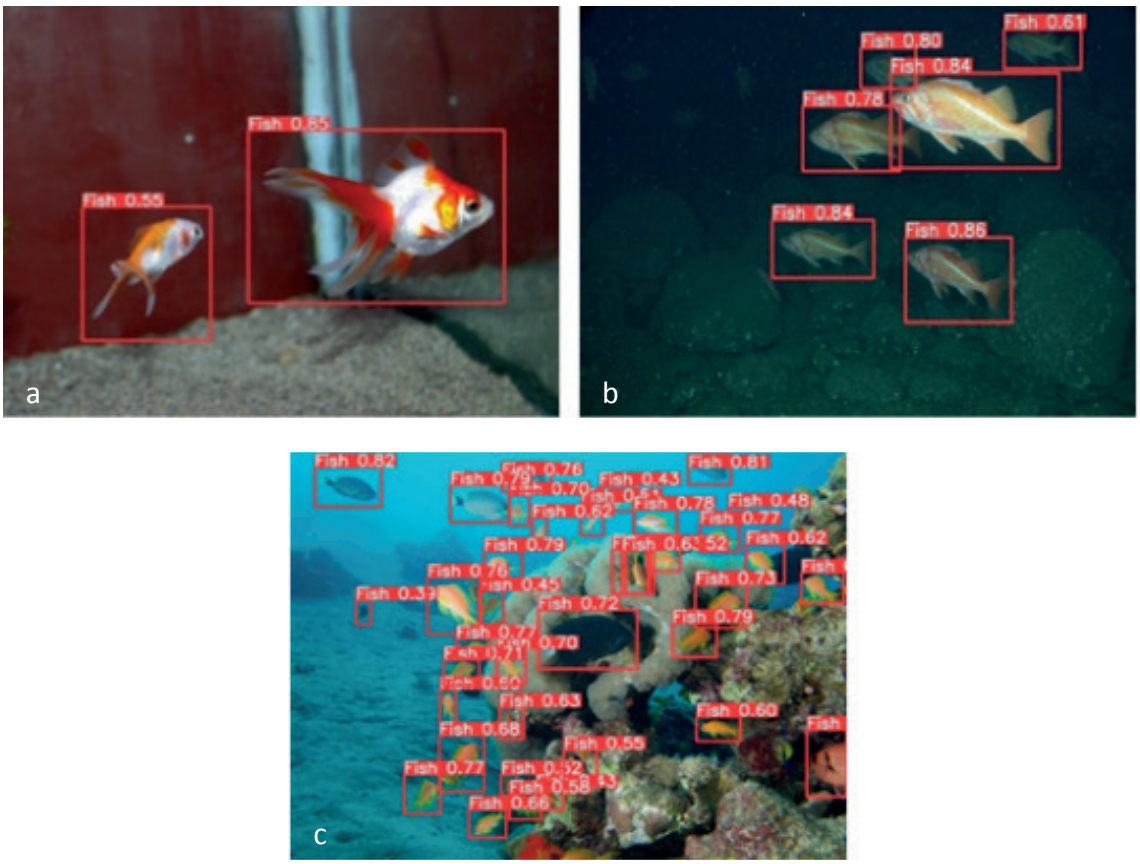

Fig. 13 - Number of fishes: less than 3 (a), from 3 to 10 (b), and more than 10 (c).

Table 3 - Accuracy under different conditions.

Sr. No.	Based on	Accuracy		
1	Number of fishes	More than 10 fishes	From 3 to 10 fishes	Less than 3 fishes
		88%	94%	98%
2	Visibility	Low	Medium	High
		90%	92%	96%
3	Similarity with background	High	Medium	Low / No
		85%	92%	97%
4	Fish size	Small	Medium	Large
		87%	91%	93%
5	Fish position	Tail View	Head View	Side View
		93%	94%	97%

2. visibility. As per the visibility of the fishes, three categories are considered: low, medium, and high.
   Visibility and accuracy are determined separately for these three categories. Fig. 14 depicts these three categories as per the visibility parameter. The second row of Table 3 indicates accuracies for the three categories based on the visibility parameters. From the table it is clear that as visibility increases, fish detection accuracy increases;

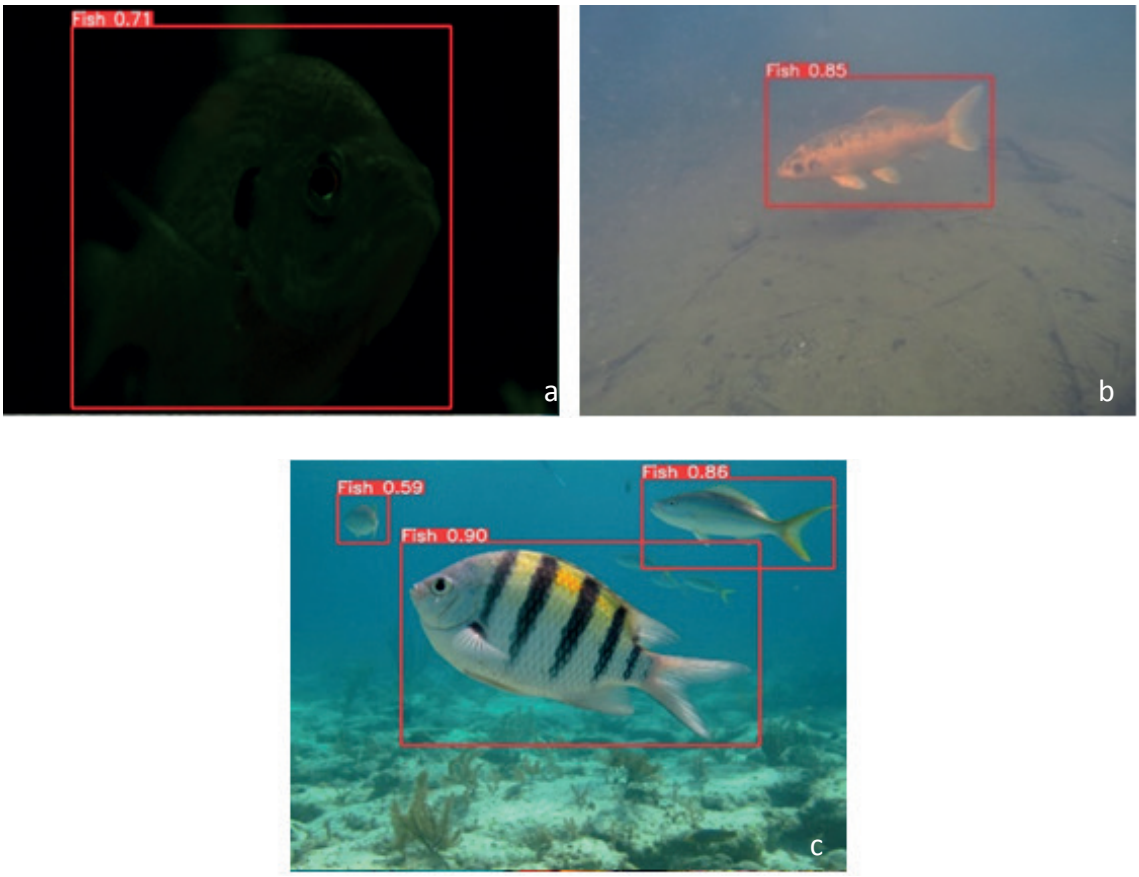

Fig. 14 - Visibility: low (a), medium (b), and high (c).

3. similarity with background. Underwater fish detection suffers from challenging backgrounds. As per the similarity of the fishes with the background, three categories are considered: low, medium, and high. An example of each category is displayed in Fig. 15. Accuracy is determined separately for these three categories as shown in the third row of Table 3. From the table it can be concluded that, as similarity with the background increases, fish detection accuracy decreases;
4. fish size. The underwater realm is inhabited by fishes of various sizes. In order to assess the performance of the model across different fish sizes, fishes were categorised into three distinct size categories: small fishes, medium fishes, and large fishes. Fig. 16 showcases representative examples of fishes belonging to each size category. The corresponding

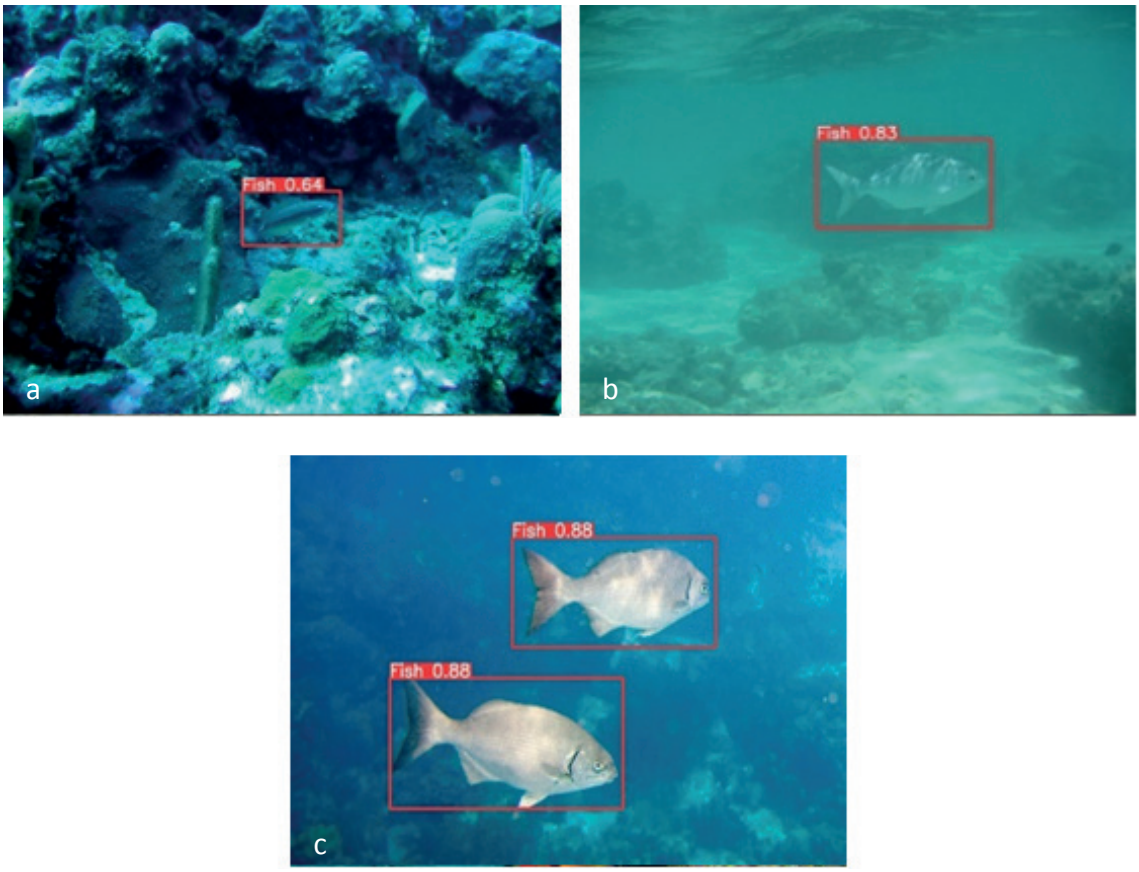

Fig. 15 - Similarity with background: high (a), medium (b), and low (c).

accuracies based on fish size are summarised in the fourth row of Table 3. The results indicate that, as fish size increases, fish detection accuracy improves;

5. fish position. The underwater environment presents fishes in a variety of positions like tail view, head view, and side view, each offering distinct visual features for detection. Fig. 17 showcases examples of fishes captured from each position category, highlighting the variations in appearance and pose. The corresponding accuracies based on fish position are presented in the fifth row of Table 3. The model demonstrates a high accuracy for side view detection and a relatively low accuracy for tail and head view detections.

4.3. Discussion

4.3.1. Overall performance

The proposed system demonstrated exceptional performance in fish detection within underwater imagery. The true positive rate of 0.91 and accuracy of 92% indicated that the system successfully identified a significant proportion of fish instances present in the images. The F1 score of 0.89 indicates a good overall performance both in terms of correctly identifying positive instances and capturing all positive instances. Recall 0.91 signifies that the model successfully

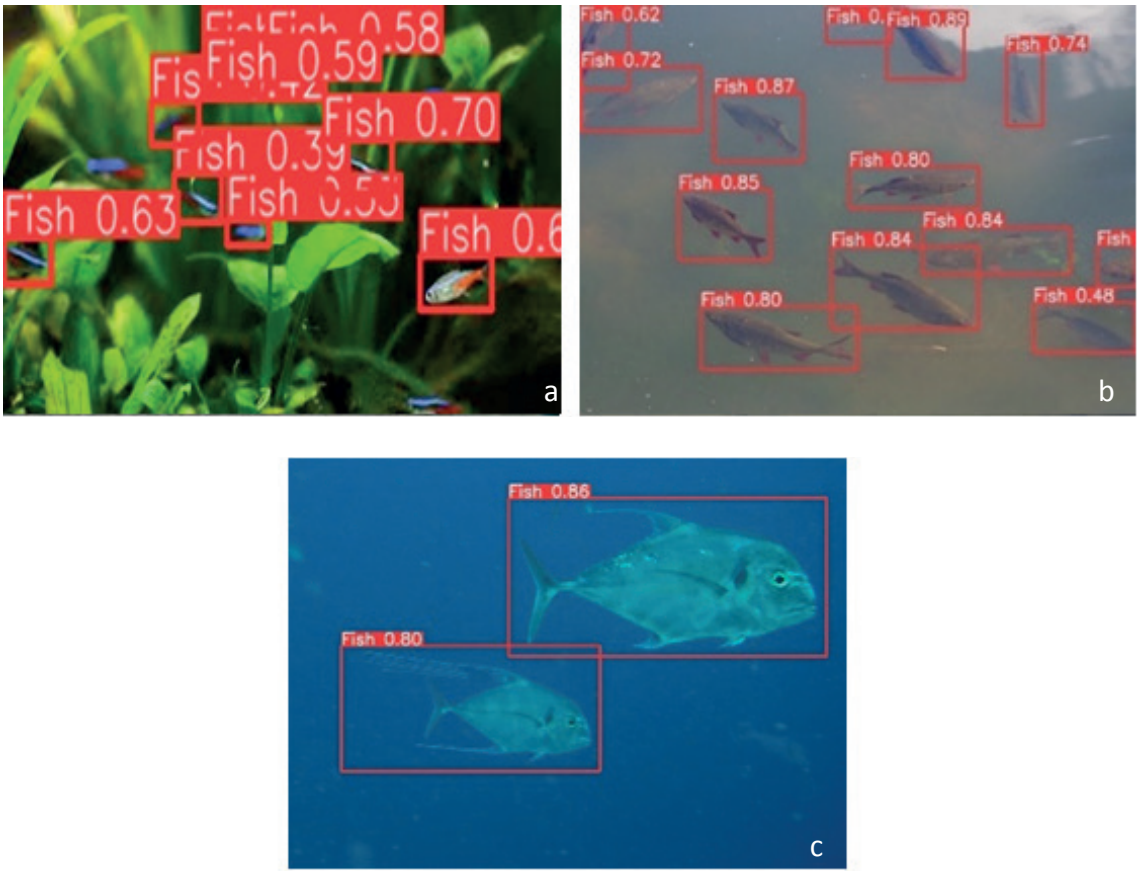

Fig. 16 - Fish size: small (a), medium (b), and large (c).

detects a high proportion of true positive instances, indicating its ability to accurately identify fishes in the underwater environment. The best confidence interval for the model is 0.382 which represents the level of uncertainty associated with the model's predictions. A lower confidence interval suggests higher confidence in the model's accuracy. The high detection accuracy of 92% can be attributed to the adaptations and modifications made to the YOLOv8 model, which effectively addressed the challenges posed by underwater environments.

4.3.2. Scenario-based performance

The accuracy analysis based on various parameters provides insightful interpretations and conclusions. As the number of fishes increases, the accuracy slightly decreases, thus suggesting that the accurate detection and localisation of individual fishes becomes more challenging in crowded scenes. To improve accuracy in scenarios with a higher number of fishes, advanced object detection techniques, such as multi-scale detection or instance segmentation, can be explored. These methods can better handle overlapping instances and improve the model's ability to accurately localise individual fishes within crowded scenes. The evaluation based on visibility conditions indicates that the model is adaptable to varying levels of visibility. Still, the model faces challenges in accurately detecting fishes in scenarios with reduced visibility, where factors such as limited contrast and blurriness impact its performance. To mitigate the challenges

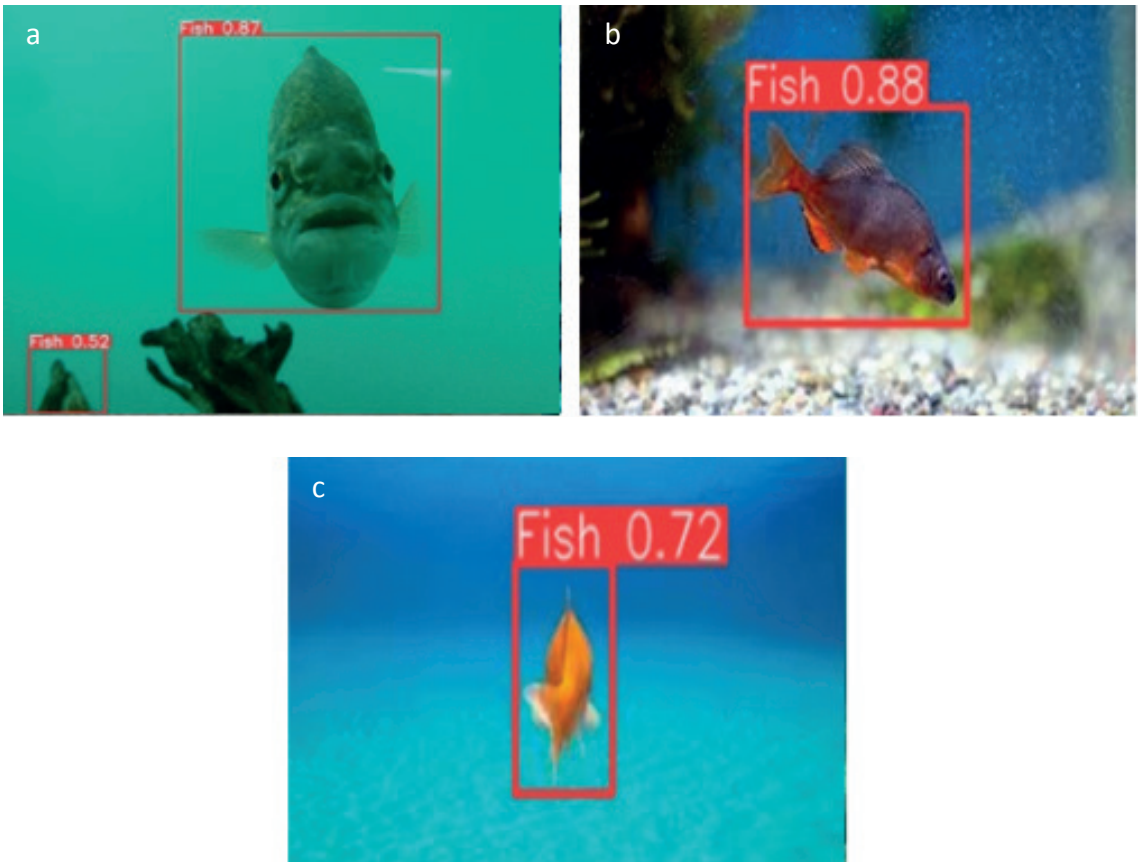

Fig. 17 - Fish position: head (a), side (b), and tail views.

posed by low visibility conditions, pre-processing techniques like image enhancement, contrast adjustment, or adaptive filtering can be applied.

The analysis based on the similarity of fishes with the background reveals the difficulty in accurately distinguishing fishes that blend with their surroundings, posing a challenge for the model's detection capabilities. Techniques such as background subtraction or foreground extraction can be employed to separate fishes from their background. By reducing the influence of background elements, the model's ability to distinguish fishes that closely resemble the background can be improved. The analysis based on fish size indicates that detecting smaller fish instances presents inherent challenges which affect the model's performance in accurately localising and classifying them. Utilising higher-resolution images or employing data augmentation techniques, specifically designed to preserve the details of small fish instances, can aid in improving their detection. The evaluation based on the position of fishes reveals that the model performs well in detecting fishes from side view but detecting fishes in head view and tail view poses challenges such as limited visibility of key distinguishing features, consequently impacting the model's performance in the detection of these instances. Augmenting the training data with more diverse side view examples and incorporating viewpoint-specific learning techniques can help the model better capture the distinguishing features and patterns associated with side view detections. Additionally, utilising additional contextual information or incorporating multi-view fusion strategies can enhance the model's ability to accurately detect

fishes from various perspectives. By addressing these gaps through a combination of advanced techniques, algorithmic improvements, and data augmentation strategies, the overall accuracy and robustness of the model can be significantly improved.

5. Comparison with existing works

It is worth noting that YOLOv8 has not been extensively utilised for underwater applications, as other methodologies have traditionally been favoured. Most existing underwater object detection methods rely on traditional computer vision techniques, such as template matching, edge detection, or feature-based methods. These approaches often require manual feature engineering and may struggle to handle complex underwater environments due to challenges such as varying illumination, water turbidity, and object occlusion. In contrast, this fish detection system based on YOLOv8 offers several advantages. Firstly, YOLO operates in real-time, enabling prompt identification of fish species. Secondly, YOLO leverages deep learning techniques, enabling end-to-end learning and the automatic learning of relevant features from the data. This eliminates the need for manual feature engineering, reducing the reliance on domain-specific knowledge and improving the adaptability of the system to different underwater environments. Furthermore, YOLOv8, being a more recent version of YOLO, incorporates various improvements and optimisations compared to its predecessors. However, it is important to note that, due to its recent introduction, there is limited existing work exploring the full potential of YOLOv8 for underwater object detection and fish species classification.

Additionally, one of the most important distinctions of this fish detection system is its ability to encompass all three essential applications: detection, tracking, and counting. While previous works have typically focused on one or two of these aspects, this model seamlessly integrates all three functionalities. This comprehensive approach provides a holistic solution for underwater fish analysis and monitoring. Moreover, this system achieves real-time performance, enabling immediate and continuous analysis of fish populations. The utilisation of the fastest model, YOLOv8n, ensures efficient processing and reduces any potential delays in capturing and processing fish-related data. By combining real-time capabilities, the integration of detection, tracking, and counting functionalities, and the utilisation of YOLOv8n, this fish detection system surpasses the limitations of previous approaches. It provides accurate and efficient fish species identification and population assessment in underwater environments.

In summary, while traditional approaches have dominated the field of underwater object detection, this work demonstrates the potential and advantages of utilising YOLO, specifically YOLOv8, for fish detection in underwater environments. By leveraging deep learning and real-time object detection capabilities, this system provides a valuable contribution to the field and opens up new avenues for underwater ecology research, fish population monitoring, and environmental conservation efforts.

6. Conclusions

In summary, the proposed bounding box-based pipeline offers an efficient and reliable solution for real-time fish detection, bounding box-based tracking, and counting, delivering high accuracy while keeping computational demands low. This makes it particularly suitable for large-scale monitoring deployments where processing speed and resource efficiency are critical.

However, bounding box-based methods may be less effective in scenarios where precise object boundaries are required, such as morphological health assessments, fine-grained behavioural studies, or biomass estimation. In such applications, alternative techniques like active contour models, snake algorithms, Mask R-CNN, or other instance segmentation methods could provide the necessary detail at the cost of higher computational overheads. As part of future work, a promising direction would be to investigate hybrid systems that use fast bounding box detection for most objects, while selectively applying high-precision segmentation to regions of interest, with the aim of balancing accuracy with efficiency.

Acknowledgments. The authors gratefully acknowledge the creators of the Google Open Images dataset V7 (Kuznetsova *et al.*, 2020). They also thank the Department of Electrical and Electronics Engineering (ECE) at Dr. Vishwanath Karad MIT World Peace University (MIT-WPU) for providing essen'al support. The authors appreciate the valuable input from anonymous reviewers, which greatly improved this research.

REFERENCES

Amraee S., Chinipardaz M. and Charoosaei M.; 2022: *Analytical study of two feature extraction methods in comparison with deep learning methods for classification of small metal objects*. Vis. Comput. Ind. Biomed. Art, 5, 13, doi: 10.1186/s42492-022-00111-6.

Barbedo J.G.A.; 2022: *A review on the use of computer vision and artificial intelligence for fish recognition, monitoring, and management*. Fishes, 7, 335.

Buono A., Catania V., Longo A. and Distante C.; 2021: *Challenges in underwater object detection: an extensive evaluation of state-of-the-art algorithms*. Sensors, 21, 1743.

Burgi K.; 2021: *Fish-bait-efficiency and benthic stock assessments using deep learning*. Mémoire de MSc. MARRES, Université Côte d'Azur, 35 pp.

Cimino M.G., Faro A.M., Mariano A.M. and Nunnari G.; 2018: *Object detection for underwater environments: a comparative analysis*. OCEANS 2018 MTS/IEEE, Charleston, SC, USA, pp. 1-8.

Coro G. and Walsh M.B.; 2021: *An intelligent and cost-effective remote underwater video device for fish size monitoring*. Ecological Informatics, 63, 101311.

CVAT Documentation; 2023: *<opencv.github.io/cvat/docs/>* (accessed 21 June 2023).

Fang X., Liu X., Li Y., Luo W. and Wang Y.; 2020: *Comparative analysis of object detection algorithms for underwater robotic applications*. Sensors, 20, 5291.

Garg R. and Phadke A.C.; 2024: *Enhancing underwater fauna monitoring: a comparative study on YOLOv4 and YOLOv8 for real-time fish detection and tracking.* In: Pandit M., Gaur M.K. and Kumar S. (eds), Artificial Intelligence and Sustainable Computing, c Algorithms for Intelligent Systems, Springer, Singapore, doi: 10.1007/978-981-97-0327-2_4.

Gruszczyński M., Rucińska A., Woźniak A. and Nowakowski K.; 2020: *Automated fish tracking and behavior analysis in a tank environment*. J. Mar. Sci. Eng., 8, 665.

He K., Zhang X., Ren S. and Sun J.; 2016: *Deep residual learning for image recognition*. In: Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 770-778, doi: 10.1109/CVPR.2016.90.

Jindal P.; 2023: *YOLOv8 is here, and it gets better!*. <pub.towardsai.net/yolov8-is-here-and-it-gets-better-54b12b87e3b9> (accessed 20 June 2023).

Joshi S. and Shivalker C.; 2023: *Why data annotation is important for machine learning and aI*. <www.habiledata.com/blog/why-data-annotation-is-important-for-machine-learning-ai/ (accessed 21 June 2023).

Khan N., Bors A.G. and Mihaylova L.; 2018: *Object detection and tracking: a review of the state-of-the-art.* Image and vision computing. 76, 1-13.

Knausgård K.M., Wiklund A., Sørdalen T.K., Halvorsen K.T., Kleiven A.R., Jiao L. and Goodwin M.; 2021: *Temperate fish detection and classification: a deep learning based approach*. Appl. Intell., 52, 6988-7001.

Kuznetsova A., Rom H., Alldrin N., Uijlings J., Krasin I., Pont-Tuset J., Kamali S., Popov S., Malloci M., Kolesnikov A., Duerig T. and Ferrari V.; 2020: *The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale.* Int. J. Comput. Vision, 128, 1956-1981, doi: 10.1007/s11263-020-01316-z.

Lekunberri X., Ruiz J., Quincoces I., Dornaika F., Arganda-Carreras I. and Fernandes J.A.; *Identification and measurement of tropical tuna species in purse seiner catches using computer vision and deep learning.* Ecol. Inf., 67, 101495. doi: 10.1016/j.ecoinf.2021.101495.

Li Y., Yu J. and Zhang Y.; 2017: *Fish counting and tracking in underwater video sequences using deep learning techniques.* Sensors, 17, 2066.

Mathworks; 2023: *Anchor Box documentation*. <in.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html> (accessed 23 June 2023)

Meimetis D., Daramouskas I., Patrinopoulou N., Lappas V. and Kostopoulos V.; 2025: *Comparative analysis of object detection models for edge devices in UAV swarms.* Mach., 13, 684, doi: 10.3390/machines13080684.

Pagire V. and Phadke A.C.; 2020: *Underwater moving object detection using GMG.* In: Proc. 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020), pp. 233-244, doi: 10.1007/978-3-030-73689-7.

Pagire V. and Phadke A.C.; 2022: *Underwater fish detection and classification using deep learning.* In: 2022 International Conference on Intelligent Controller and Computing for Smart Power, Hyderabad, India, pp. 1-4, doi: 10.1109/ICICCSP53532.2022.9862410.

Pagire V. and Phadke A.C.; 2024: *Fish Detection by Hybrid of MOG2 and LSBP Methods.* Nonlinear Optics, 60, No. 1-2, 137-157, ISSN 15430537.

Priyadarshni D., Kolekar M.H.; 2020: *Underwater Object Detection and Tracking.* In: Pant M., Sharma T., Verma O., Singla R., Sikander A. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 1053. Springer, Singapore, pp. 837–846, doi: https://doi.org/10.1007/978-981-15-0751-9_76.

Qin Y., Wu C., Yu N. and Zhang L.; 2020: *Robust visual tracking of underwater fish using deep learning*. IEEE Access, 8, 55090-55100.

Rath S. and Gupta V.; 2023: *Performance comparison of YOLO object detection models – An intensive study*. <learnopencv.com/performance-comparison-of-yolo-models> (accessed 20 June 2023).

Rath S.; 2023: *YOLOv8 Ultralytics: State-of-the-Art YOLO models.* <learnopencv.com/ultralytics-yolov8/> (accessed 27 August 2023).

Redmon J. and Farhadi A.; 2017: *YOLO9000: better, faster, stronger*. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

Redmon J. and Farhadi A.; 2018: *YOLOv3: an incremental improvement.* In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, pp. 7794-7803.

Redmon J., Divvala S., Girshick R. and Farhadi A; 2016: *You only look once: unified, real-time object detection*. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 779-788.

Ren S., He K., Girshick R. and Sun J.; 2016: *Faster R-CNN: towards real-time object detection with region proposal networks.* Adv. Neural Inf. Process. Syst., 28, 91-99, doi: 10.48550/arXiv.1506.01497.

Sah S., Shringi A., Ptucha R., Burry A. and Loce R.; 2017: *Video redaction: a survey and comparison of enabling technologies.* J. Electron. Imaging, 26, 051406, doi: 10.1117/1.JEI.26.5.051406.

Skalski P.; 2023: *Track and count using YOLOv8*. <blog.roboflow.com/yolov8-tracking-and-counting/> (accessed 24 June 2023).

Sprague M., Ramsey D. and Taylor G.; 2014: *Traditional field methods for studying fish populations.* Fish. Tech. 3rd ed., 55-105.

Ultralytics documentation; 2023: <*docs.ultralytics.com/models/yolov8/*> (accessed 28 August 2023).

Yousif K.A. and Ghareeb A.A.; 2019: *Real-Time fish detection and tracking in underwater videos*. J. Mar. Sci. Eng., 7, 188.

Zhang Y., Jia K., Liu L. and Li Y.; 2016: *Fish detection and tracking for underwater visual sensor networks*. Sensors, 16, 927.

Corresponding author: Anuradha Phadke
MIT-WPU: Dr Vishwanath Karad MIT World Peace University
Survey No, 124, Paud Rd, Kothrud, Pune, Maharashtra 411038, India
Phone: +91 020 30273400; e-mail: anuradha.phadke@mitwpu.edu.in